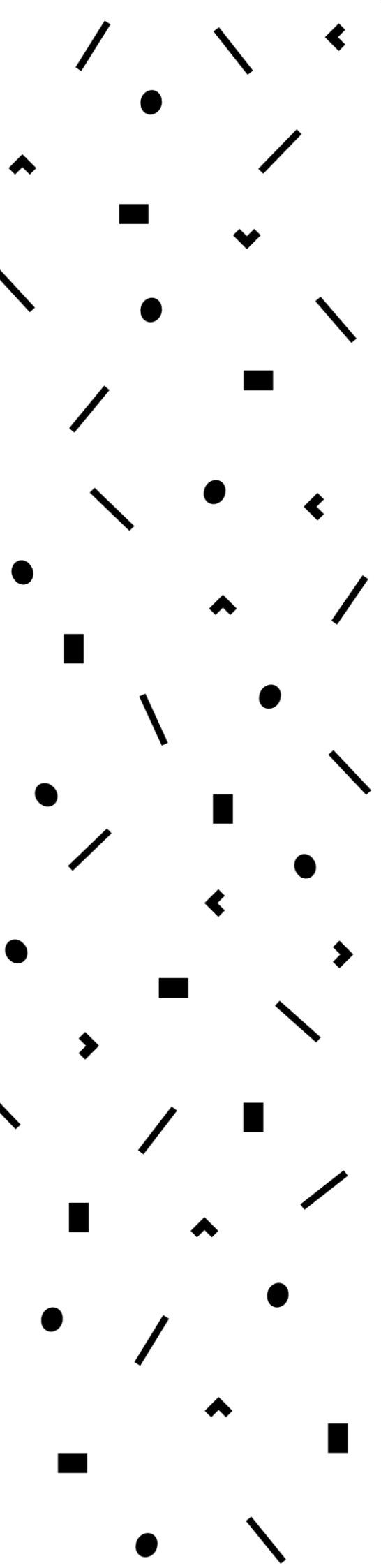


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Esineiden internetin soveltaminen Raspberry Pin avulla

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Tietokone-elektroniikka
Opinnäytetyö
Syksy 2017
Riku Roivainen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

ROIVAINEN, RIKU

Esineiden internetin soveltaminen Raspberry Pin avulla

Tietokone-elektroniikan opinnäytetyö, 32 sivua, 3 liitesivua

Syksy 2017

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli luoda käytännön esimerkkejä esineiden internetin käyttämisestä Raspberry Pi Model B+ -tietokoneen avulla.

Työssä ohjattin ja tuotiin tietoa Raspberryyn liitetyistä komponenteista. Opinnäytetyö esittelee, kuinka Raspberryä voidaan ohjata internetin välityksellä käyttäjän selaimen kautta sekä automaattisen sähköposti-ilmoituksen ohjelmoinnin. Opinnäytetyössä esitellään työssä käytetyt komponentit ja ohjelmat sekä selostetaan niiden toiminta. Ohjelmistokielinä toimivat Python, HTML ja JavaScript. Opinnäytetyö perustuu Tornadon ja Dataplicityn lähdekoodeihin.

Opinnäytetyön lopputulos on kaksi toimivaa esimerkkiä esineiden internetin käytöstä Raspberry Pin avulla. Myös tietoturvan merkitystä pohdittiin.

Asiasanat: esineiden internet, ohjelmointi, python, html, javascript, raspberry pi

Lahti University of Applied Sciences
Degree Programme in Information Technology

ROIVAINEN, RIKU

Applying Internet of Things using Raspberry Pi

Bachelor's Thesis in computer electronics, 32 pages, 3 pages of appendices

Autumn 2017

ABSTRACT

Objective of the thesis was to create practical examples on how to use the Internet of Things using the Raspberry Pi Model B+ computer.

In this thesis, electronical components were used to communicate to and from Raspberry Pi. The thesis presents how Raspberry Pi can be controlled over the internet via the user's web-browser and programming of an automatic e-mail notification. The thesis presents the components and programs used and explains how they function. Software languages used were Python, HTML and JavaScript. The codes used in this thesis were based on the source codes of Tornado and Dataplicity.

As a result of the thesis, two working examples of using the Internet of Things with the Raspberry Pi were created. The significance of information security was also considered.

Keywords: IoT, programming, python, html, javascript, raspberry pi

SISÄLLYS

LYHENNELUETTELO

GPIO	General Purpose Input/Output, yleis sisään/ulostulo
HTML	Hypertext Markup Language, hypertekstin merkintäkieli
HTTP	Hypertext Transfer Protocol, hypertekstin siirtoprotokolla
IoT	Internet of Things, esineiden internet
IP	Internet Protocol, internetin protokollaosoite
SMTP	Simple Mail Transfer Protocol, sähköpostin lähetys-protokolla

1 JOHDANTO

Raspberry Pi on Linux-pohjainen sulautettu järjestelmä, jonka ominaisuuksiin kuuluu pieni koko ja vähäinen virrankulutus. Sen kaikki komponentit ja liitännät on rakennettu yhdelle piirilevylle. Laitetta voi käyttää useilla eri käyttöjärjestelmillä ja ohjelmoida useilla ohjelmointikielillä. Tässä opinnäytetyössä käytettiin Raspian-käyttöjärjestelmää ja Python, JavaScript ja HTML-ohjelmointikieltä.

Esineiden internet on yleistettynä laitteiden kytkemistä verkkoon, mutta sitä käytetään yhä useammin kuvaamaan laitteita, jotka keskustelevat internetin välityksellä jokapäiväisen elämän helpottamiseksi. Se antaa myös laitteille mahdollisuuden kommunikoida erilaisten verkkotyyppien välityksellä. Tietoa yleensä kerätään laitteiden käytöstä tai niiden lähiympäristöstä ja tämä tieto viedään edelleen muihin laitteisiin. Päätin tehdä opinnäytetyön aiheesta esineiden internetin kasvavan suosion takia.

Opinnäytetyön tavoitteena oli käyttää Raspberry Pi model 2 B+-tietokonetta ja sen GPIO-pinnejä (liite 1) luomalla ohjelmia, jotka käyttävät hyväksi internetiä ja mahdollistavat kommunikoinnin internetin välityksellä. Kommunikointia varten Raspberryyn liitettiin komponentteja, joita ohjattiin internetin välityksellä tai vastaanotettiin tietoa tarvittaessa.

2 TOTEUTUSTAVAT

2.1 Työn lähtötilanne

Työn tavoitteena oli saada yhteys Raspberryyn internetin välityksellä ja näin hyöty käyttää esineiden internetiä. Toiminta-alue rajoitettiin noin 20 neliön huonetilaan, jossa Raspberry olisi yhteydessä modeemiin ja tätä kautta internetiin.

Ongelmana oli löytää sopiva protokolla kommunikointiin Raspberryn kanssa. Esineiden internetiä varten on ajan myötä kehitetty useita eri protokollia, joista jokaisella on omat ominaisuutensa ja näin soveltuvat eri tarkoituksiin. Toteutuksessa harkittiin android-pohjaista puhelinsovellusta, mutta tämä konsepti hylättiin helpomman ja yksinkertaisemman HTML-pohjaiseen sovellukseen. Tämä mahdollistaisi GPIO-pinnien käytön laitteesta riippumatta ja useimmilla selaimilla. Näin ohjelman käyttöä ei rajoitettaisi turhaan.

Ensimmäisen esimerkin toteutustavaksi päädyttiin GPIO-pinnien ohjaukseen ja komponenteiksi valittiin kaksi eriväristä lediä, punainen ja vihreä. Raspberry PI:n aktiivisen GPIO-pinnin ulostulojännite on 3,3 V. Ledien kynnysjännitteet ovat molemmilla 2,6 V, vastuksien yli tuleva jännite olisi silloin 0,7 V. Maksimivirrat olivat 90 mA punaiselle ledille ja 120 mA vihreälle. Ledien eteen sijoitettavien vastusten resistanssi mitataan kaavalla 1:

$$R = U/I$$

0,7 V / 90 mA = ~8 ohmia

0,7 V/ 120 mA = ~6 ohmia

Pienimmät vastukset käytössäni olivat 220 ohmia, joilla ledit syttyivät ja paloivat ilman ongelmia.

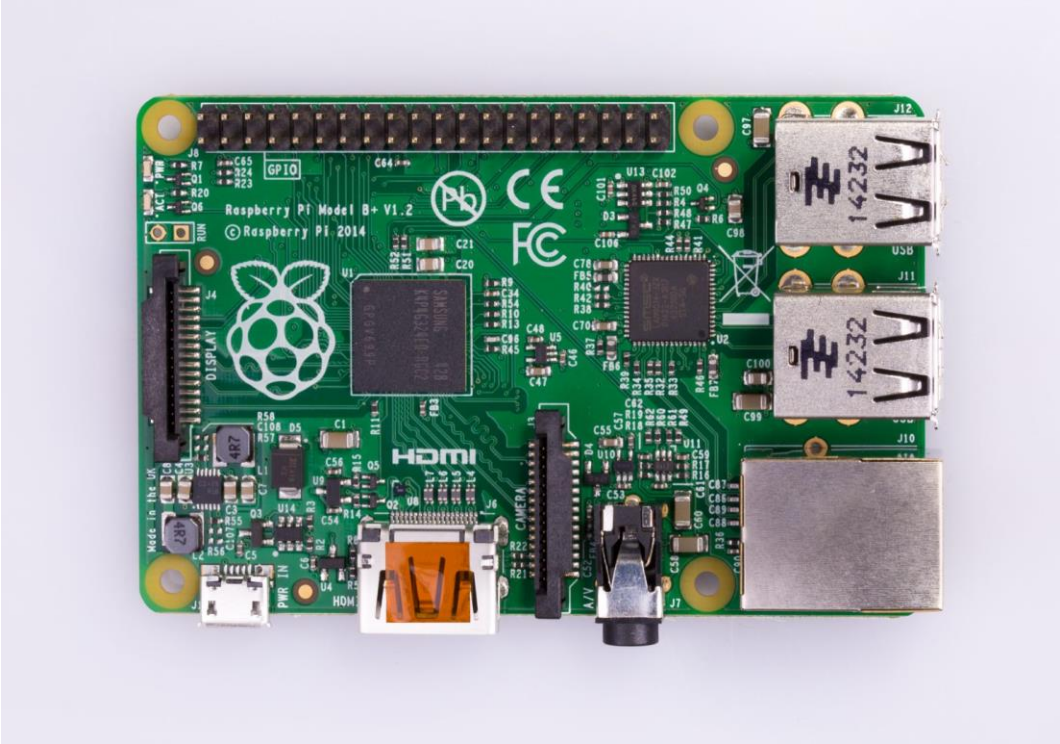
Ensimmäisen esimerkin toteutukseen valittiin Tornado framework, joka on avoin Python-pohjainen versio Facebookin omistamasta FriendFeed-palvelusta. (Tornado 2017.)

Toinen esimerkki saatiin toteutettua pelkästään Python ohjelmalla, joka lähettäisi sähköpostin oven avautuessa, toimien alkeellisena murtohälyttimenä. Komponentteina toimivat painokytkin ja Littlefuse 59045 reed-rele, joka pysyy suljettuna magneettikentän lähellä. (liite 2) Magneettikentän muodostamiseen käytettiin kestomagneettia.

2.2 Raspberry Pi

Raspberry Pi on yhdelle piirilevylle rakennettu tietokone, jonka Eban Upton, Rob Mullins, Jack Lang ja Alan Mycroft kehittivät alun perin opetuskäyttöön (Raspberry Pi Foundation 2017). Raspberryn etuja ovat pieni koko ja alhainen virrankulutus. Ajan myötä sen harrastajayhteisö on kasvanut huomattavasti ja on nykyään suosittu alusta erilaisille projekteille.

Raspberry Pi:stä on saatavilla useita eri malleja. Niiden hinnat ja komponentit vaihtelevat, suorituskyky ja muistin koko ovat parantuneet uusimmissa malleissa. Raspberryssä on ohjelmoitavia GPIO-pinnejä, yleiskäytettäviä portteja. Näistä pinneistä osa voivat toimia älykkäämmin, vaikka I2C-tiedonsiirtoa varten. Työssä käytetyn mallin kopio näkyy kuviossa 1. Työssä käytettiin Raspberryn käyttöjärjestelmänä Raspbian nimistä Debianiin perustuvaa käyttöjärjestelmää. Ohjelmointiin käytettiin Raspberryn integroitua kehitysympäristöä, IDLEä.



KUVIO 1. Raspberry Pi B+ (Raspberry Pi 2017)

2.3 Tornado

Web-palvelimena toimi Python-pohjainen Tornado, joka on tarkoitettu reaaliaikaisten web-sivustojen alustaksi. Se pystyy vastaanottamaan useita samanaikaisia yhteyksiä ja on erinomainen sovelluksille, jotka tarvitsevat pitkäkestoista yhteyttä palvelimen ja sovelluksen välille. Tornadolla on useita ominaisuuksia, kuten käyttäjän todennus ja evästeet, joilla Facebook halusi helpottaa reaaliaikaisten ympäristöjen kehitystä.

Tornado voidaan jakaa neljään osaan: Web framework, joka vastaa pyyntöjen käsittelystä, asiakkaan ja palvelimen HTTP-sovellukset, asynkroninen verkkokirjasto, joka sisältää HTTP:n luokat ja erillinen coroutine-kirjasto, jolla sallitaan asynkronisen koodin kirjoittaminen. (Tornado 2017.)

Päätin käyttää Tornadoa sen laajan dokumentoinnin ja suosion ohjelmointiyhteisöjen keskuudessa. Tornadon esimerkkikoodi on saatavilla kotisivullaan tai GitHub-verkkosivustolla.

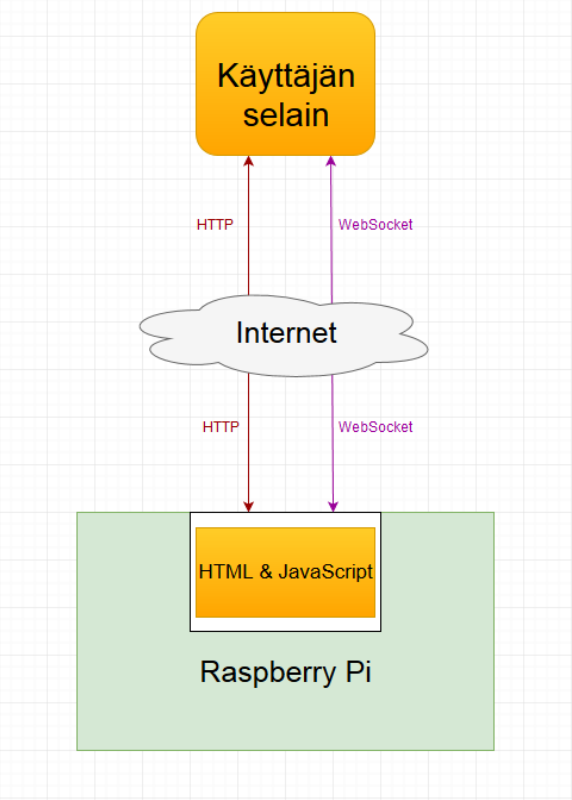
2.4 WebSocket

Tarvitsin keinon luoda pysyvän yhteyden, joka voi tukea käyttäjän ja palvelimen käynnistämiä tapahtumia GPIO-pinnien hallitsemiseksi. WebSocket mahdollisti tämän.

WebSocket-yhteyden muodostaminen tunnetaan nimellä handshake, kättely, joka on esitetty kuviossa 2. Prosessi alkaa, kun käyttäjä lähettää HTTP-pyynnön palvelimelle. Tähän pyyntöön sisältyy päivitysotsikko, joka ilmoittaa palvelimelle, että asiakas haluaa luoda WebSocket-yhteyden. HTTP siis määrittelee, miten web-selain ja web-palvelin viestivät keskenään. (Korpela 2009.)

Jos palvelin tukee WebSocket-protokollaa, se suostuu kättelyyn. Kättelyn valmistuessa alkuperäinen HTTP-yhteys korvataan WebSocket-yhteydellä. Nyt jompikumpi osapuoli voi aloittaa jatkuvan tiedon lähettämisen.

WebSocket ei ole ainoa olemassa oleva tapa kaksisuuntaisen viestinnän tarjoamiseksi asiakkaan ja palvelimen välillä, mutta sopii työn toteutukseen, sillä useimmat selaimet tukevat WebSocket-protokollaa (liite 3).



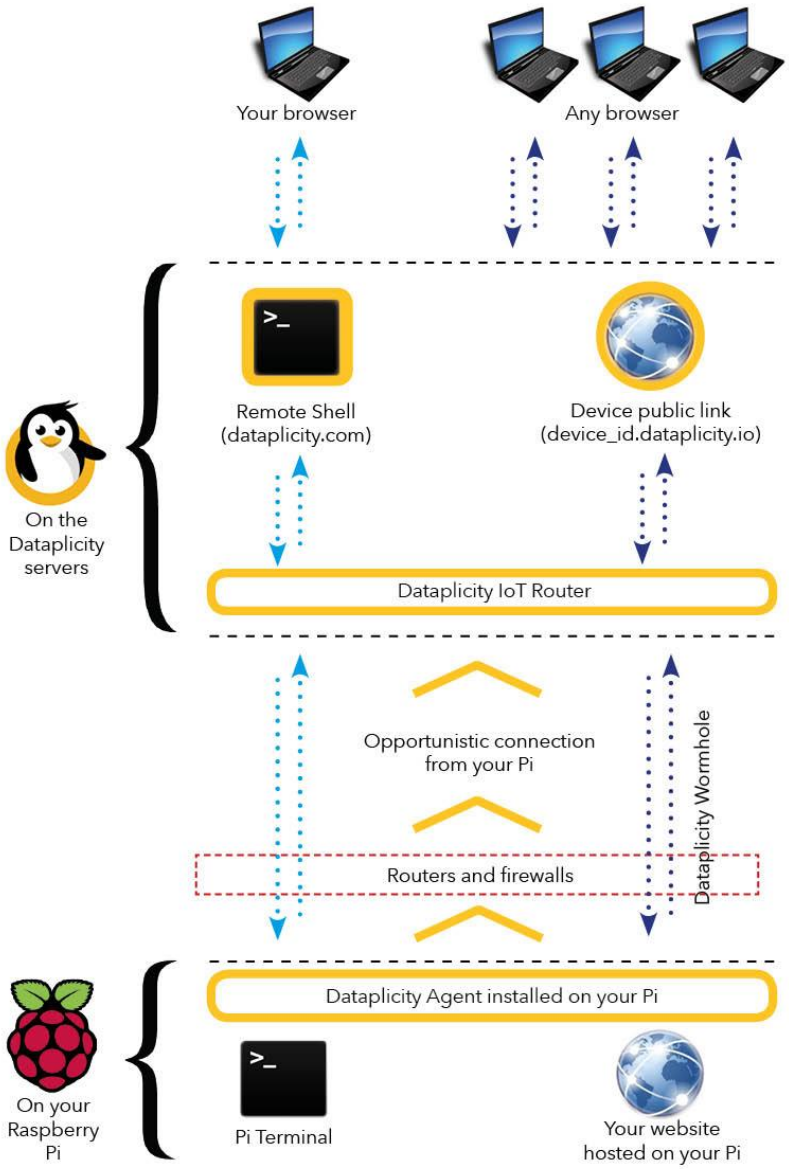
KUVIO 2. HTTP:n ja WebSocketin avulla luotu yhteys

2.5 Dataplicity

Dataplicity on verkkopalvelu ja sovellus, joka on kehitetty mahdollistamaan etäyhteys Raspberry Pihin.

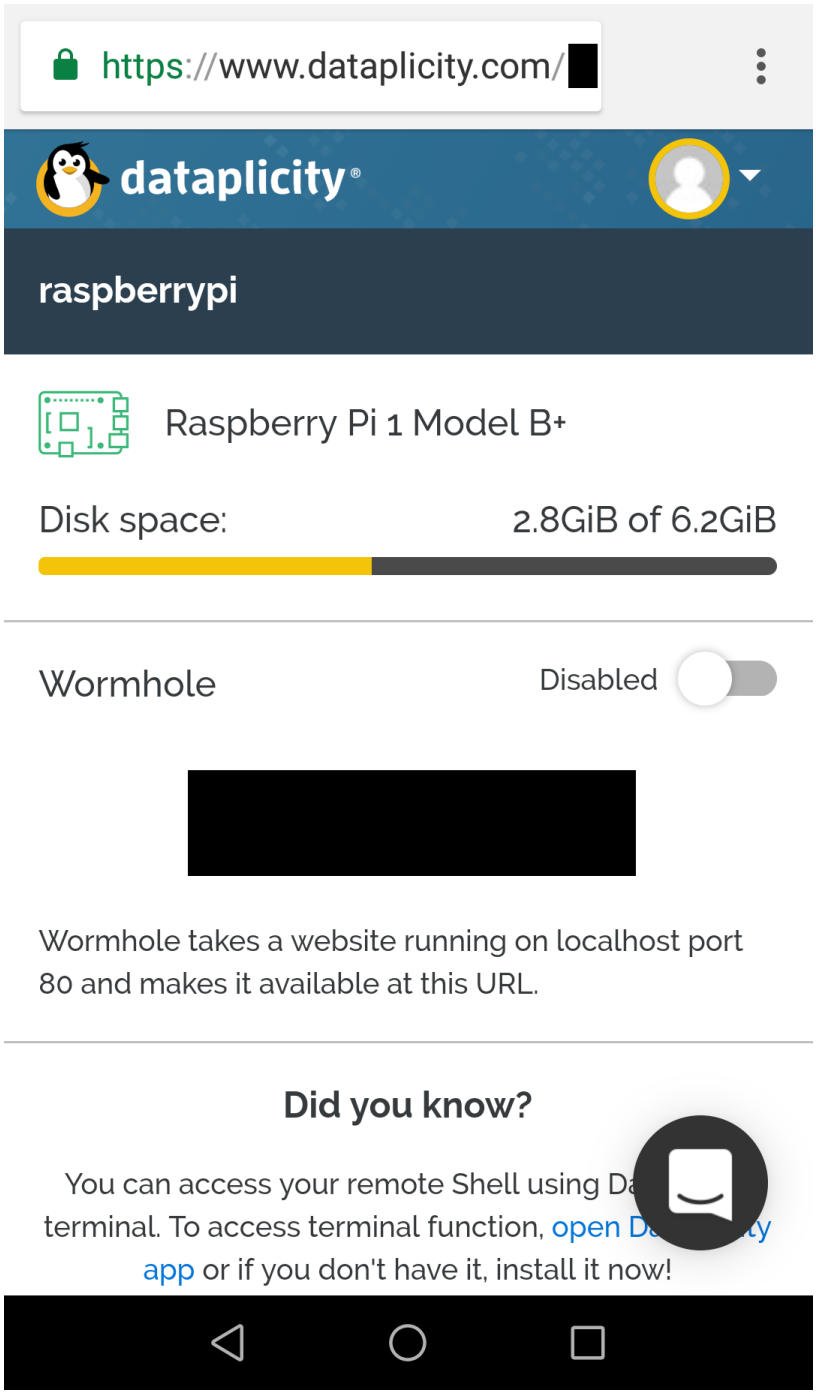
Dataplicity mahdollistaa yksinkertaisen yhteyden Raspberry Pihin ilman porttiohjausta, verkon manuaalista konfigurointia tai IP-osoitteiden määrittelyä. Käytännössä tämä tarkoittaa, että voit käyttää Dataplicityn laitteita missä tahansa niillä on internetyhteys. (Dataplicity 2017.)

Dataplicityn kotisivulla on mahdollista nähdä yhdistetty Raspberry ja avata wormhole, madonreikä. Madonreiän avulla on mahdollista hallita Raspberry Pillä olevaa verkkosivustoa ilman ylimääräisiä asetuksia. Tämän toiminta on selostettu kuviossa 3. Samalla saadaan myös uniikki osoite, jonka syöttämällä selaimeen saadaan yhteys Raspberryllä isännöimään sivustoon kuten kuvassa 1 näkyy. Raspberry voidaan myös käynnistää uudelleen tarvittaessa.



KUVIO 3. Dataplicityn toimintaperiaate (Dataplicity 2017)

Dataplicity on ilmainen ensimmäiselle yhdistetylle laitteelle, tämän jälkeen 2 dollaria kuukaudessa laitetta kohti. Android ja iOS-sovellukset ovat saatavilla mobiililaitteiden omilla sisältökaupoilla. (Dataplicity 2017.)

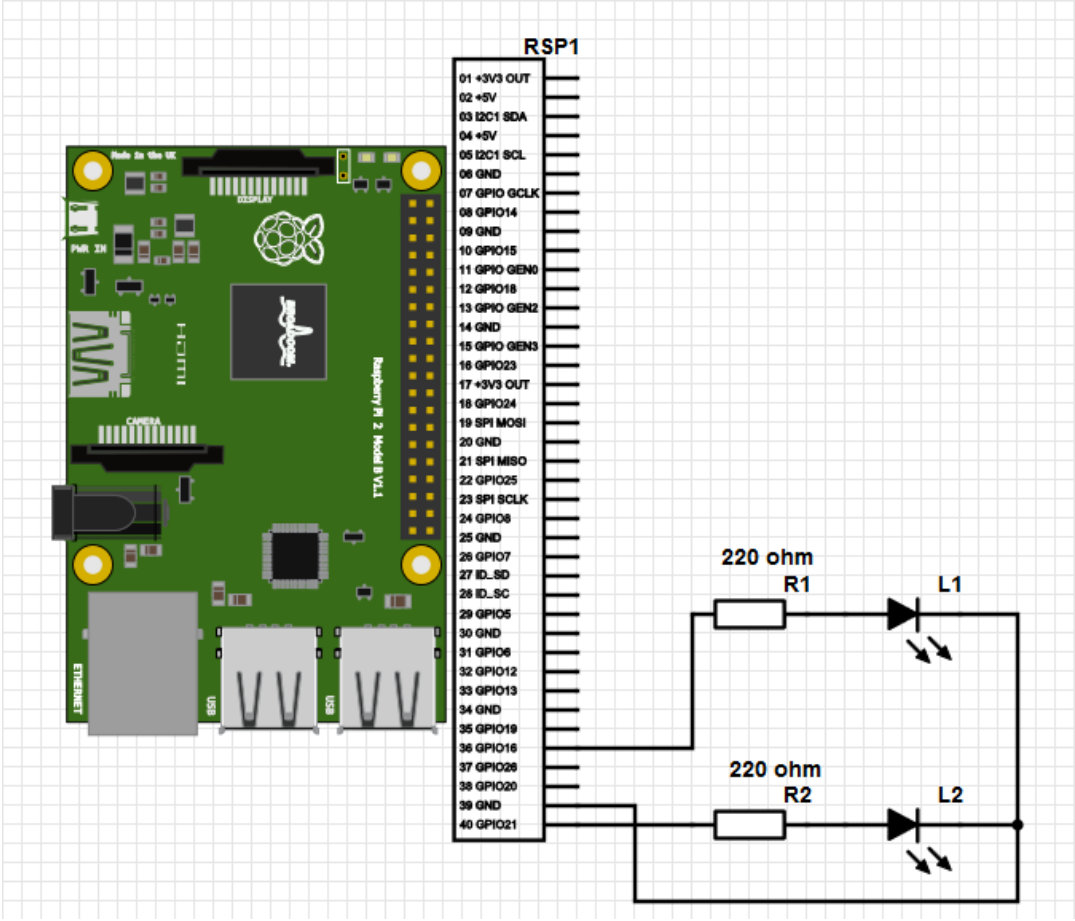


KUVA 1. Työn Raspberry näkyvillä Dataplicityn mobiilisivulla

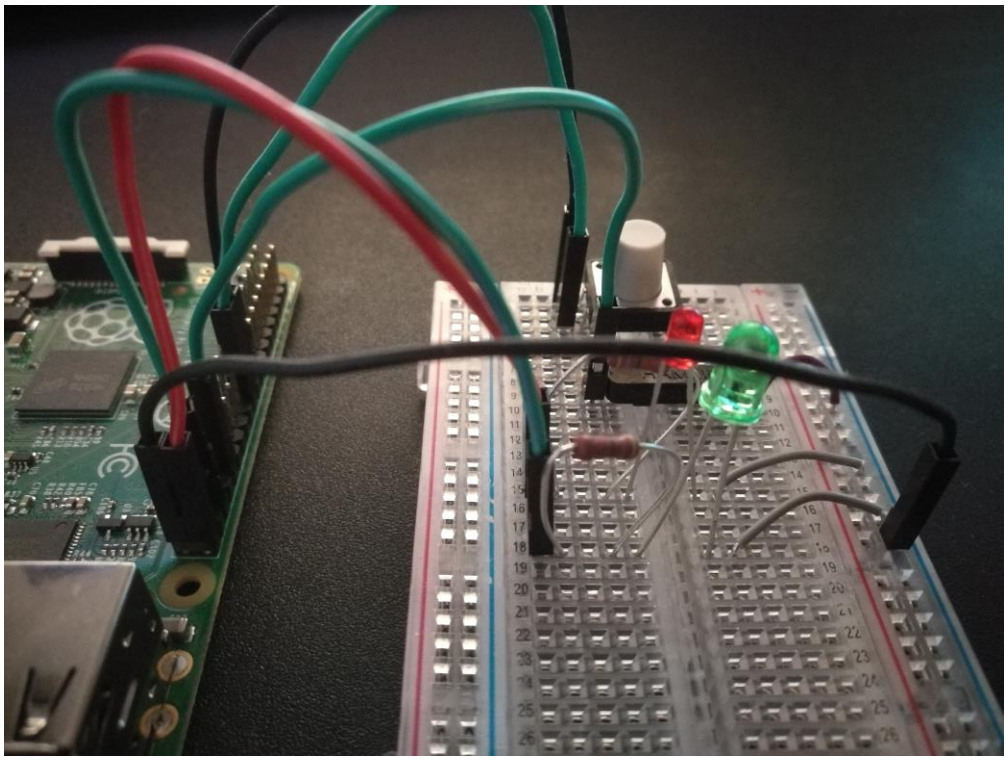
3 OSA 1: LEDIEN OHJAUS SELAIMEN KAUTTA

Ensimmäisessä esimerkissä pyrittiin yksinkertaiseen toteutukseen. Tavoitteena oli käyttää älypuhelinta vaihtamaan Raspberry Pi:n tilaa muuttamalla GPIO-pinnejä tilojen 0 (ei aktiivinen) ja 1 (aktiivinen) välillä. Komponenteiksi valittiin kaksi eriväristä lediä. Ohjelman testaus suoritettiin kokoamalla kuvion 4 mukainen koekytkentä, jonka näemme kuvassa 2.

Ongelmana oli sopivan toteutustavan löytäminen. Android-pohjaista sovellusta yritettiin mutta hylättiin yksinkertaisemman toteutustavan vuoksi. HTML-pohjainen sovellus oli mahdollinen, jos ohjelman avulla voisimme lähettää aktivointikäskyt GPIO-pinneille. Raspberry Pin piti siis pystyä jakamaan internetin välityksellä HTML-dokumentti, jonka avulla voitaisiin hallita Raspberryn GPIO-pinnejä. Etuna tässä olisi ohjelman käytön mahdollisuus alustasta riippumatta, olettaen että niillä pääsisi internetiin. Tarvitsin silti keinon päästä käsiksi HTML-sivuun. Tässä kohtaa Dataplicity antoi mahdollisuuden relatiivisen helppoon etäkäyttöön. Vaikka dokumentaatiota muista tavoista löytyi, antoi Dataplicity tuloksia hyvin nopeasti ja vähällä vaivalla. Toteutukseen käytettiin Tornadon ja Dataplicityn lähdekoodeja.



KUVIO 4. Osa 1 piirikaavio

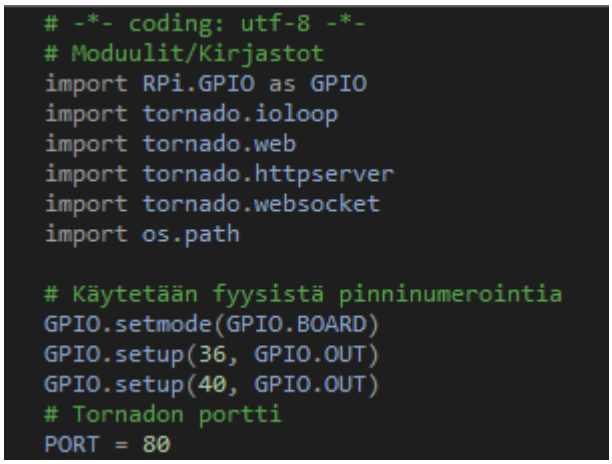


KUVA 2. Osa 1 koekytkentälevyllä

3.1 Python

Python-ohjelmointikielenä soveltuu hyvin niin alkeelliseen kuin vaativaankin ohjelmointiin ja on saatavilla ilmaiseksi monille järjestelmille. Pythonin ominaisuuksiin kuuluu helppo luettavuus, joka mahdollistaa ohjelmoijan keskittymään itse ohjelmointiin yksityiskohdista huolehtimisen sijaan.

Työssä käytettiin Pythonin versiota 3. Eroa aikaisempaan versioon 2 ovat tietyt yhteensopimattomat avainsanat ja ominaisuudet. Yksi huomattavin, vaikkakin pieni muutos on print-komennon, tulostus-syntaksin muutos. Python 2 tulostuslausuma on korvattu print () -funktiolla, mikä tarkoittaa, että ohjelmoijan tarvitsee kääriä tulostettava objekti sulkeisiin.



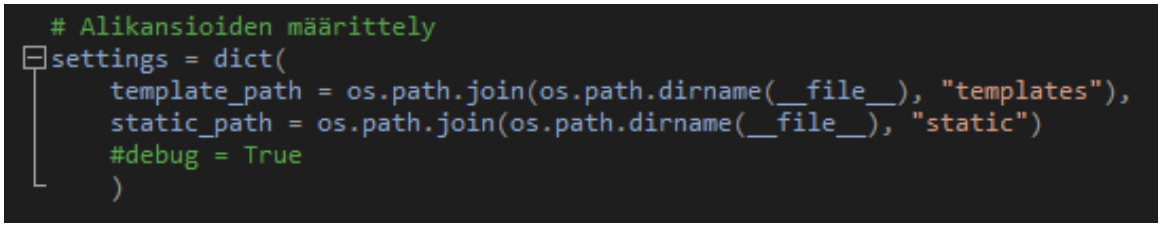
```
# -*- coding: utf-8 -*-
# Moduulit/Kirjastot
import RPi.GPIO as GPIO
import tornado.ioloop
import tornado.web
import tornado.httpserver
import tornado.websocket
import os.path

# Käytetään fyysistä pinninumerointia
GPIO.setmode(GPIO.BOARD)
GPIO.setup(36, GPIO.OUT)
GPIO.setup(40, GPIO.OUT)
# Tornadon portti
PORT = 80
```

KUVA 3. Osa 1 Python ohjelmakoodi (1)

Kuvassa 3 tuodaan ohjelmalle välttämättömät kirjastot ja moduulit. RPi.GPIO-moduuli tulee valmiina Raspianin mukana ja vastaa GPIO-pinnien toiminnasta. Tornadon tarvitsemat moduulit tulee ladata erikseen.

Aluksi määritellään, mitä pinnejä ohjelma käyttää. Käytössä ovat piirilevyn fyysisen numeroinnin mukaan pinnit 36 ja 40. Ledit on kytketty maahan käyttämällä pinniä 39, mutta tätä ei tarvitse ilmoittaa ohjelmassa.

Myös portti 80 määritellään, jota WebSocket protokolla käyttää salaamattomille yhteyksille.



```
# Alikansioiden määrittely
settings = dict(
    template_path = os.path.join(os.path.dirname(__file__), "templates"),
    static_path = os.path.join(os.path.dirname(__file__), "static")
    #debug = True
    )
```

KUVA 4. Osa 1 Python ohjelmakoodi (2)

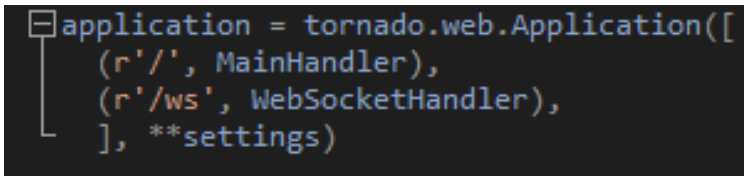
Ohjelma määritellään käyttämään templates- ja static-alikansioita kuvassa 4. Templates-kansio sisältää ohjelman HTML-osuuden nimeltään index.html ja static puolestaan JavaScriptin, ws-client.js. Näihin tiedostonimiin viitataan vasta myöhemmin. Debug on jäämiä ohjelmoinnin työstövaiheesta, ja ohjelmalla ei ole tarvetta sille käytön aikana.

```
class MainHandler(tornado.web.RequestHandler):
  def get(self):
      self.render("index.html")

class WebSocketHandler(tornado.websocket.WebSocketHandler):
  def open(self):
      print("Yhteys toimii.")

  def on_message(self, message):
    print ('Painoit:'), message
    if message == "punainen_LED_ON":
      GPIO.output(40, True)
    if message == "punainen_LED_OFF":
      GPIO.output(40, False)
    if message == "vihrea_LED_ON":
      GPIO.output(36, True)
    if message == "vihrea_LED_OFF":
      GPIO.output(36, False)
```

KUVA 5. Osa 1 Python ohjelmakoodi (3)

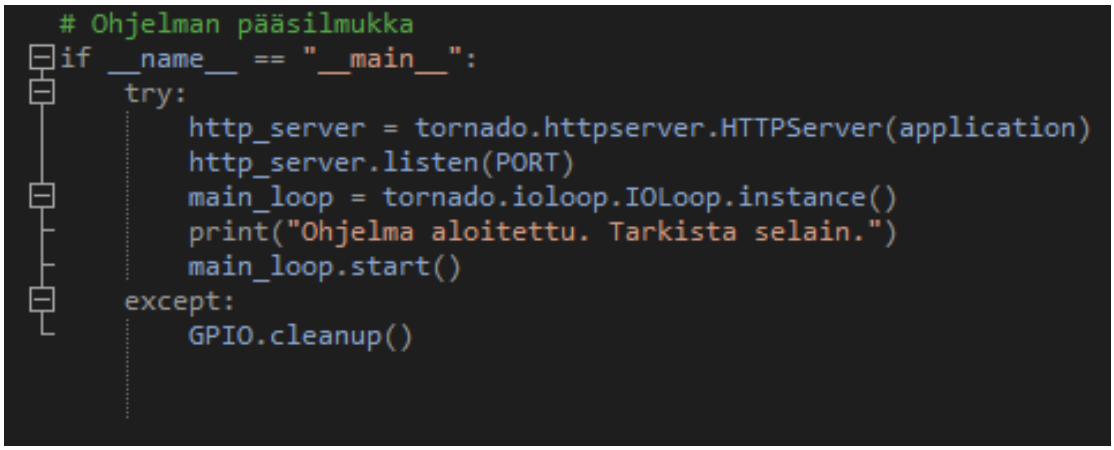
Kuvassa 5 luodaan kaksi tapahtumakäsittelijäluokkaa. Nämä ovat ohjelmiston toimintoja, jotka suoritetaan vastauksena tapahtumaan. Ensimmäistä käytetään noutamaan MainHandlerissa luotu ohjelman HTML-osuus nimeltään index.html. WebSocketHandlerissa taas avataan WebSocket-yhteys ja tulostetaan viesti Raspberryn terminaaliin kertomaan, kun yhteys on muodostettu.

Käskyn saamisen jälkeen etäyhteyttä käyttämällä tulostetaan vahvistusviesti Raspberryn terminaaliin ja vaihdetaan GPIO-pinnien tilaa vastaavasti.



KUVA 6. Osa 1 Python ohjelmakoodi (4)

Kuvassa 6 kutsutaan edellä luotuja luokkia ja alikansioiden määritystä, jotta ohjelma tietää missä HTML- ja JavaScript-ohjelmat sijaitsevat.



KUVA 7. Osa 1 Python ohjelmakoodi (5)

Kuvassa 7, ohjelman pääsilmukassa, aloitetaan hyväksymään yhteydet portilta 80 ja näin ohjelma pystyy lähettämään, sekä vastaanottamaan dataa. Ohjelman aloituksesta tulostetaan viestin Raspberryn terminaaliin, mutta tämä ei vielä tarkoita, että yhteys on muodostettu onnistuneesti. Tämä silmukka mahdollistaa ohjelman ajon, kunnes käyttäjä lopettaa sen manuaalisesti Raspberryn terminaalilta tai Dataplicityn antaman selaimen kautta. Jälkimmäinen keino vaatii toimivan yhteyden Raspberryyn. GPIO.cleanup palauttaa käytettävät portit takaisin syöttötilaan kun ohjelma lopetetaan.

3.2 JavaScript

JavaScript on tekstipohjainen, dynaaminen ohjelmointikieli jota käytetään yleensä osana web-pohjaista sovellusta, usein HTML-ohjelman lisäosana, Se on kuitenkin ohjelmointikielenä yleiskäyttöinen ja sitä voi käyttää suorittamaan tavallisia laskentoja.

JavaScript mahdollistaa HTML-ohjelmien luonnin, jotka toimivat vuorovaikutuksessa käyttäjän kanssa, ohjaavat selainta ja tarvittaessa luovat HTML-sisältöä. JavaScript yksinään ei kuitenkaan voi tuottaa grafiikkaa, mutta se kykenee muotoilemaan HTML-sivustoja. Monet verkkoselaimet tukevat JavaScriptia, vaikka useimmat antavat mahdollisuuden estää sen käytön. Tämän takia sivuja ei kannata tehdä vain JavaScriptin varassa toimiviksi.

Tässä työssä JavaScript mahdollistaa verkkosivun painikkeiden käytön joilla ohjaamme GPIO-pinnien kautta Raspberryyn kiinnitettyjä ledejä lähettämällä vahvistusviestin jonka Python-ohjelma vastaanottaa ja vaihtaa pinnien tilaa vastaavasti.

```
$(document).ready(function(){

    var WEBSOCKET_ROUTE = "/ws";

    if(window.location.protocol == "http:"){
        var ws = new WebSocket("ws://" + window.location.host + WEBSOCKET_ROUTE);
        }
    else if(window.location.protocol == "https:"){
        var ws = new WebSocket("wss://" + window.location.host + WEBSOCKET_ROUTE);
        }

    ws.onopen = function(evt) {
        $("#ws-status").html("Online");
        };

    ws.onclose = function(evt) {
        $("#ws-status").html("Offline");
        };

    $("#vihrea_on").click(function(){
        ws.send("vihrea_LED_ON");
        });

    $("#vihrea_off").click(function(){
        ws.send("vihrea_LED_OFF");
        });

    $("#punainen_on").click(function(){
        ws.send("punainen_LED_ON");
        });

    $("#punainen_off").click(function(){
        ws.send("punainen_LED_OFF");
        });

});
```

KUVA 8. Osa 1 JavaScript ohjelmakoodi

Kuvassa 8 luodaan WebSocket-yhteys ja siihen kiinnitetään funktiot, joilla pystytään ohjaamaan ledejä. Yhteyden tila määritellään callback-funktioiden avulla, ja näkyy näin sivustolla. Javascript toimii näin selaimen taustalla ja havaitsee verkkosivun painikkeiden käytön.

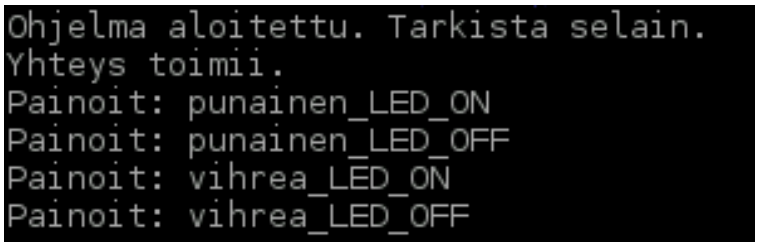
3.3 HTML

```
<html>
<head>
 <title> Lediohjaus 1.01 </title>
</head>
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
<body>

<div>
 <h1 style="background-color:LightGray;text-align:center">Ledien kontrollointi</h1>
</div>

 <div id="ws-status" style="text-align:center;">...</div>
 <h1 style="color:red;text-align:center;font-size:200%">Punainen</h1>
<p style="text-align:center;">
 <input type="button" id="punainen_on"  value="ON" style="height:50px; width:200px">
 <input type="button" id="punainen_off" value="OFF" style="height:50px; width:200px">
</p style="text-align:center;">
<h1 style="color:green;text-align:center;font-size:200%">Vihre&auml</h1>
<p style="text-align:center;">
 <input type="button" id="vihrea_on"  value="ON" style="height:50px; width:200px">
 <input type="button" id="vihrea_off" value="OFF" style="height:50px; width:200px">
</p>

 <script src="//ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
 <script src="{{ static_url("ws-client.js") }}"></script>
 </body>
 </html>
```

KUVA 9. Osa 1 HTML ohjelmakoodi

Kuvan 9 HTML-ohjelmassa määritellään sivuston ulkoasu. Se sisältää ledien ON- ja OFF-painikkeet, joita painamalla ledit syttyvät ja sammuvat. JavaScript-osio sijoitetaan script-tagien väliin. Ohjelmassa haetaan myös jQuery, JavaScript-kirjasto, sekä määritetään alikansiossa erillään olevan JavaScriptin tekstitiedoston nimeksi ws-client.js.

Painikkeet välittävät tiedon edelleen Raspberryn Python-ohjelmaan JavaScriptin avulla. Kuvassa 10 näemme Raspberryn terminaalin tulosteen ohjelman aloituksesta ja yhteyden muodostamisesta sekä vahvistuksen jokaisen painalluksen jälkeen.



KUVA 10. Osa 1 näkymä Raspberryn terminaalissa

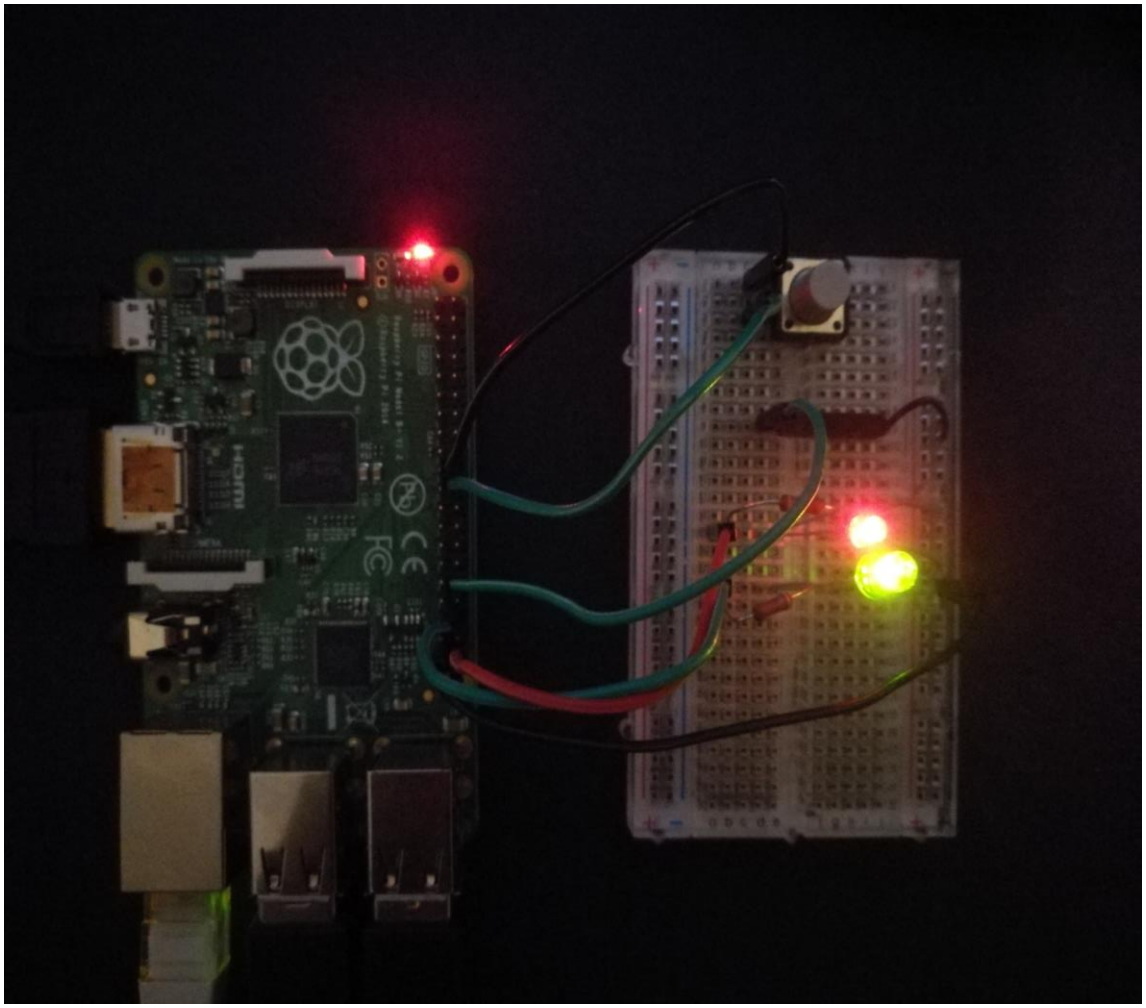
3.4 Valmis ohjelma

Ohjelman kokonaisuus koostuu kolmesta kooditiedostosta: ledit.py, index.html ja ws-client.js. Kuvassa 11 näkyy, kuinka käynnistämisen jälkeen on mahdollista käyttää sitä selaimelta Dataplicityltä saadun verkko-osoitteen avulla.

Wormhole voidaan tarvittaessa avata ja sulkea Dataplicityn sivulla, mutta tämä ei estä ohjelmaa jatkamasta, jos se on käynnissä, vain yhteys Raspberryyn katkeaa. Raspberryä voidaan hallita Dataplicityn sivulla ja tarvittaessa käynnistää sen uudelleen, mutta ohjelmaa ei ole määritetty aloitettavaksi käynnistyksen yhteydessä automaattisesti. Jos GPIO-pinnin tila vastaa jo saatavaa komentoa, viesti tulostuu Raspberryn terminaaliin mutta pinnin tila ei vaihdu.

KUVA 11. Valmis sivusto älypuhelimen selaimelta katsottuna

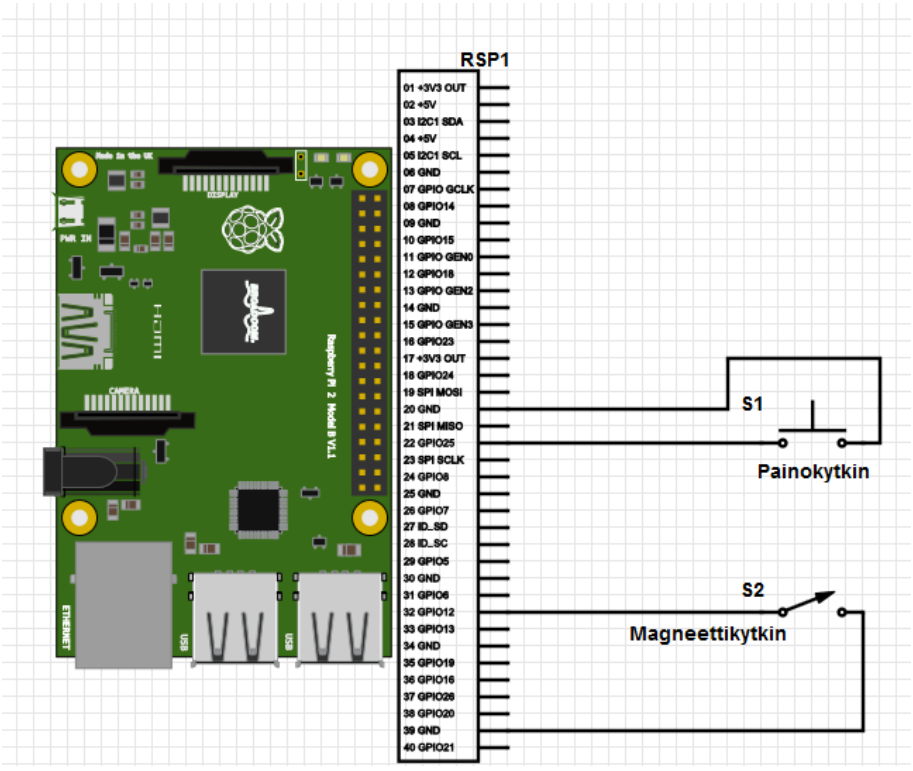
Ledit reagoivat selaimelta syötettyihin käskyihin odotusten mukaisesti kuvassa 12.



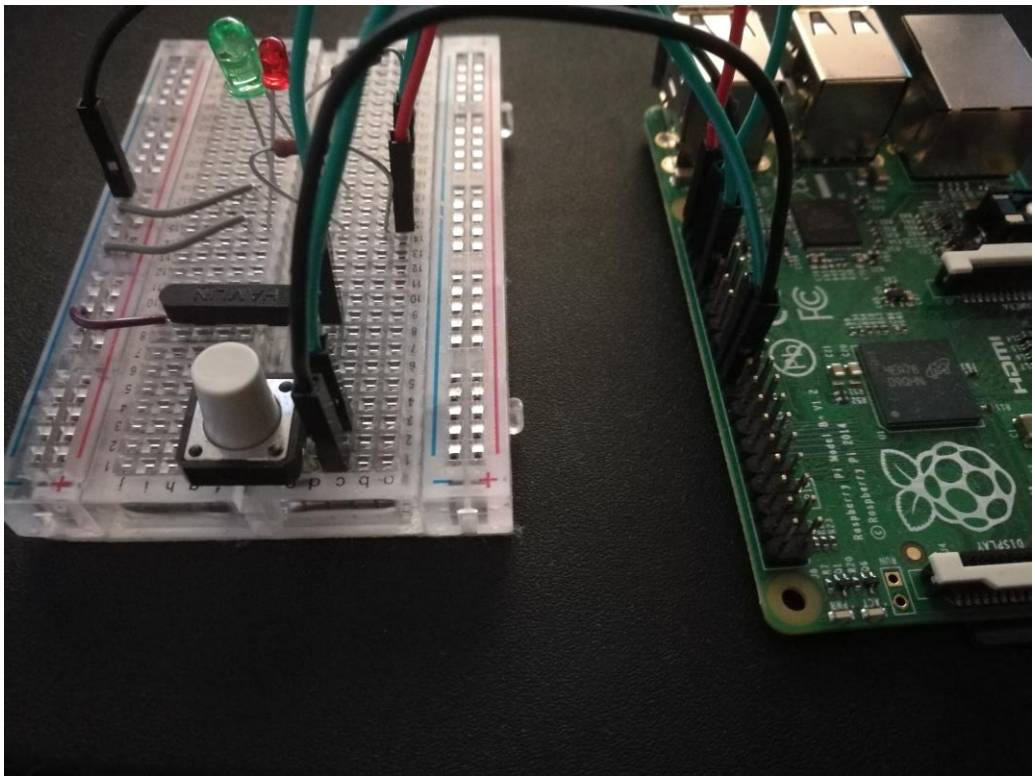
KUVA 12. Koekytkentälevyllä syttyneet ledit

4 OSA 2: AUTOMAATTINEN ILMOITUS

Ensimmäisen esimerkin jälkeen halusin toteuttaa ohjelman joka pystyisi lähettämään sähköpostin magneettisensorin tilan muuttuessa. Tämä toimisi alkeellisena murtohälyttimenä. Toteutuksen piirikaavio on esitetty kuviossa 5 ja koekytkentä kuvassa 13. HTML-pohjainen toteutus ei soveltunut työn vaatimukseen. Käyttäjän kannalta on hankala pitää silmällä verkkosivua, jossa ilmotettaisiin onko ovea avattu vai ei. Halusin nopean vastauksen oven avauksen tapahtuessa, ja sähköpostin lähetys sopi tähän. Työssä käytetty magneettikytkin tuli olla kosketusetäisyydellä kestomagneettiin. Testaus pystyttiin suorittamaan yksinkertaisesti pitämällä magneetti kiinni kytkimessä. Kytkimen herkkyydessä oli parantamisen varaa, mutta täytti työn vaatimukset. Erillinen painokytkin lisättiin, jotta hälytyksen käynnistäminen ei olisi pelkästään ohjelman ajamisesta riippuva, helpottaen käyttöprosessia. Työ olisi mahdollista suorittaa halutessa ilman erillistä kytkintä.



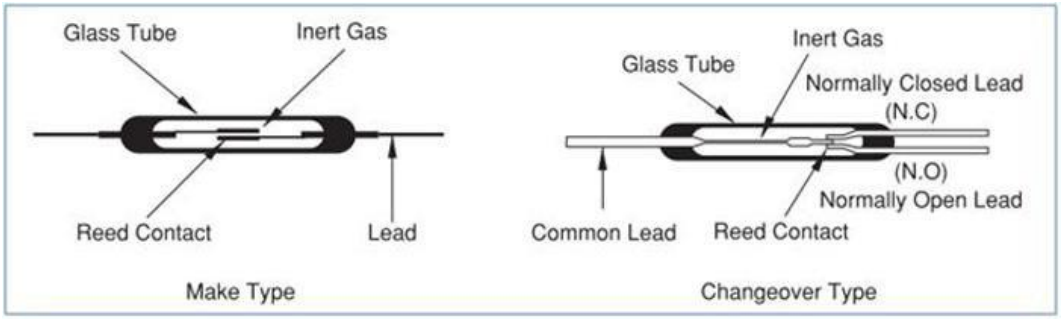
KUVIO 5. Osa 2 piirikaavio



KUVA 13. Osa 2:n magneettikytkin ja painokytkin

4.1 Magneettikytkin

Magneettinen kytkin koostuu reed-releestä, sähköisesti ohjattava kytkimestä, joka on valettu koteloon (tyhjiöön). Rele koostuu kahdesta kosketinkielestä siten, että niiden väliin jää rako. Magneettikenttään joutuessaan kosketinkielet avautuvat tai sulkeutuvat muodostaen niihin liittyvän virtapiirin. Reed-rele voi olla avautuva (NC, Normally Closed) tai sulkeutuva (NO, Normally Open), joka määrää kuinka se reagoi magneettikenttään. Työssä käytetty kytkin on sulkeutuva ja sen sulkeutuminen magneetin läheisyydessä mahdollistaa ohjelman viestin lähetyksen aktivoitumisen. Kytkimen toiminta on selostettu kuviossa 6.



KUVIO 6. Reed-kytkimen toiminta (Engineers Garage 2017)

4.2 Python

Kuvassa 14 tuodaan ohjelmalle välttämättömät kirjastot ja moduulit kuten aiemmassa esimerkissä, mutta tämän esimerkin suorittaminen ei vaadi ulkoisten kirjastojen tai moduulien lataamista.

```
# -*- coding: utf-8 -*-
# Moduulit/kirjastot
import smtplib
import time
import datetime
import RPi.GPIO as GPIO
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
# Käytetään fyysistä pinninumerointia
GPIO.setmode(GPIO.BOARD)
# Määritetään kytkimen ja sensorin tila
GPIO.setup(32 , GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(22, GPIO.IN,pull_up_down=GPIO.PUD_UP)
```

KUVA 14. Osa 2 Python ohjelmakoodi (1)

Työssä tarvitaan kaksi GPIO-pinniä, magneetti- ja painokytkimelle. Käytössä ovat GPIO-pinnit 32 ja 22, joiden tilat määritetään ykköseksi (high). Maana toimivat pinnit 39 magneettikytkimelle ja 20 painokytkimelle.

```
def alarm(channel):
  timestamp = time.time()
  stamp = datetime.datetime.fromtimestamp(timestamp).strftime('%H:%M:%S')
  if GPIO.input(channel):
    # Magneetti irti sensorista - ovi on aukaistu
    print("Ovi auki klo " + stamp)
    message = "Ovi aukaistu."
    msg = MIMEText(message)
    msg['subject'] = 'Raspberry Pi'
    msg['from'] = 'lähettäjän sähköpostiosoite'
    msg['to'] = 'vastaanottajan sähköpostiosoite'
    #lahetys
    s = smtplib.SMTP('smtp.gmail.com','587')
    s.ehlo()
    s.starttls()
    s.ehlo()
    s.login('lähettäjän sähköpostiosoite','sähköpostin salasana')
    s.sendmail(msg['from'], msg['to'], msg.as_string())
    s.quit()
    time.sleep(10)
  else:
    # Ovi suljettu avauksen jälkeen.
    print("Ovi kiinni klo " + stamp)
```

KUVA 15. Osa 2 Python ohjelmakoodi (2)

Sähköposti lähetetään ohjelmassa määritettyyn sähköpostiin kuvassa 15. Seuraavaksi määritetään viestin otsikko, vastaanottajan sähköposti ja viestin sisältö. Myös lähettäjän sähköpostiin kirjautuminen suoritetaan ohjelmassa.

Pythonissä on valmiiksi asennettuna kirjasto, jonka avulla voidaan muodostaa yhteys SMTP-palvelimeen, tässä tapauksessa Googlen Gmailiin. Yhteys käyttää porttia 587.

```
def main():

  try:
    while True :
      time.sleep(0.5)

  except KeyboardInterrupt:
    GPIO.cleanup()

print("Ohjelma aloitettu. ALARM OFF.")

# Kytkimen aktivointi
while True:
    inputValue = GPIO.input(22)
    if (inputValue == False):
        print("20 sek. aikaa poistua.")
        time.sleep(20)
        print("ALARM ON")
        # Hälytys aktivoitu
        GPIO.add_event_detect(32, GPIO.BOTH, callback=alarm, bouncetime=200)
        time.sleep(0.5)
```

KUVA 16. Osa 2 Python ohjelmakoodi (3)

Ohjelma alkaa painokytkintä painamalla, jonka jälkeen ohjelma odottaa kaksikymmentä sekuntia ennen jatkamista. Tämän jälkeen aktivoidaan varsinainen hälytys. Jos ovi aukeaa, ohjelma pitää kymmenen sekunnin tauon sähköpostin lähetyksen jälkeen, ja jatkaa toimintaansa, ellei ohjelmaa keskeytetä erikseen Raspberrystä. Dataplicityä pystytään käyttämään Raspberryn uudelleenkäynnistyksessä sekä tarvittaessa lopettamaan ohjelma. Ajan arvot on määritelty itselleni sopiviksi, oven avaus hälytyksen aktivoinnin jälkeen lähettää sähköpostin riippumatta kuka oven avaa, tai mihin aikaan. Kuvassa 16 näemme tämän osan ohjelmasta, oven aukaisu ja kellonaika jäävät ylös Raspberryn terminaaliin kuvassa 17.

```
Ohjelma aloitettu. ALARM OFF.
20 sek. aikaa poistua.
ALARM ON
Ovi auki klo 14:08:39
Ovi kiinni klo 14:08:52
```

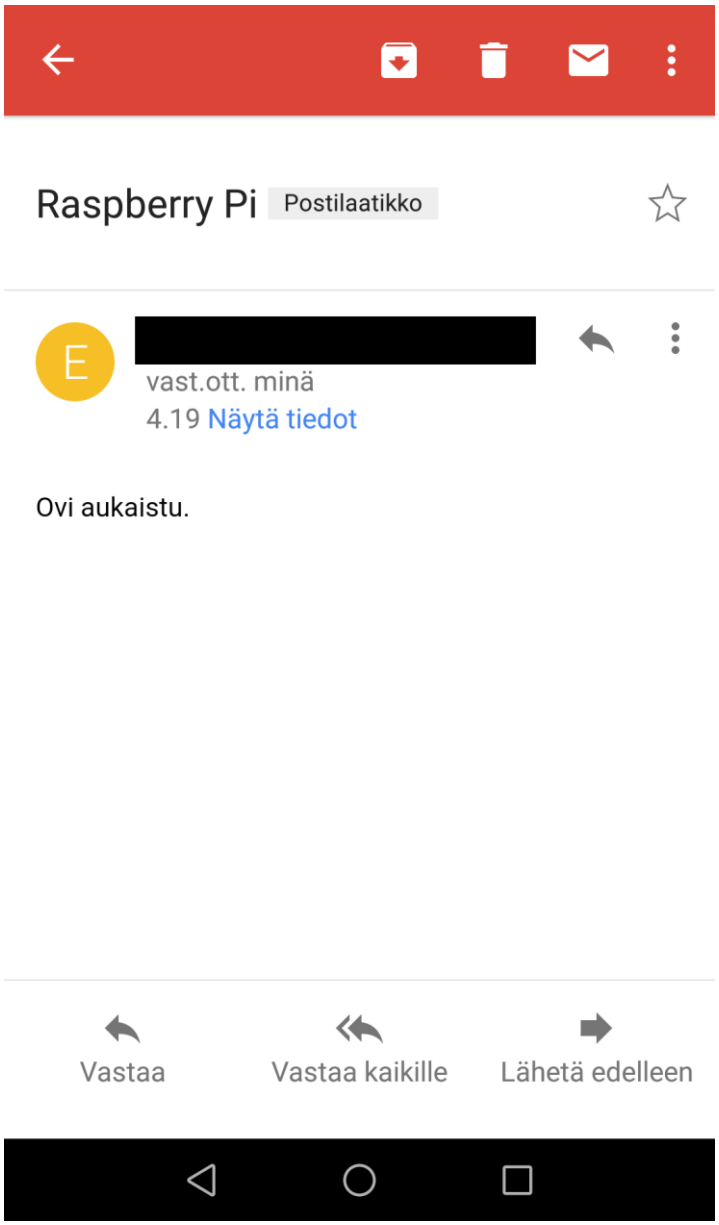
KUVA 17. Osa 2 näkymä Raspberryn terminaalissa

4.3 Valmis ohjelma

Raspberry on valmis havaitsemaan oven avaamisen. Saadun viestin voi lukea selaimelta tai älypuhelimen Gmail-sovelluksesta. Kellonaikaa ei tarvittu liittää sähköpostiviestiin, sillä Gmail ilmoittaa viestin saapumisajan automaattisesti. Itse viestin sisällöllä ei ole väliä, ja voitaisiin tarvittaessa jättää tyhjäksi. Kuvassa 18 näemme ohjelman tuloksen. Ohjelmaa ei ole määritetty aloitettavaksi käynnistyksen yhteydessä automaattisesti.

Sähköpostin vastaanottamiseksi Gmaililla, tulee vastaanottajan ensin hyväksyä vähemmän turvallisten sovelluksien käyttö (Google 2017).

Jos haluaisimme lähettää sähköpostin mukana liitteitä, tarvitsisimme erillisen moduulin ja tiedoston liitettäväksi. Jos ohjelmaa haluttaisiin työstää lisää, voisi vaikkapa liikesensorin ja kameran liittää erillisiin GPIO-pinneihin ja vaatia niiden aktivoinnin ennen sähköpostin lähetystä.



KUVA 18. Sähköpostiviesti älypuhelimen Gmail-sovelluksessa

5 YHTEENVETO

Raspberry Pi oli entuudestaan tuttu alusta ja ohjelmisto-osaamista oli kertynyt edellisten projektien kautta. Toteutus vaati silti tutkimustyötä ja materiaalia oli runsaasti tutkittavaksi.

Ohjelmien toteutus oli osaltaan helppo Tornadon ja Dataplicityn lähdekoodien ja laajan dokumentoinnin avulla. Aikaisempi Python osaaminen helpotti myös ohjelmointia. Varsinainen haaste oli ymmärtää ohjelman kulku tietoliikenteen osalta, sillä sen kannalta ei minulla ollut paljoa aiempaa kokemusta. Varsinkin ensimmäisestä osasta tuli monipuolinen kokonaisuus, josta riitti haastetta. Toteutustapoja vaikutti olevan useampia, ja näiden testaaminen vei aikaa.

HTML- ja JavaScript-osuudet olivat pienempiä verrattuna Pythoniin, mutta aikaa kului niidenkin käyttöönotossa. Näitä ohjelmointikieliä olen käyttänyt vähemmän viime vuosien aikana ja olin tyytyväinen päästessäni harjoittamaan muistiani niiden osalta.

Pidän harmillisena, että toisessa esimerkissä jouduin kirjoittamaan sähköpostin kirjautumistiedot itse ohjelmassa. Itse koodia ei lähetetä sähköpostin mukana, mutta salasanan kirjoittaminen ylös on silti turvallisuusriski. Voimme yrittää salata sen koodaamalla, mutta on myös annettava tapa purkaa salaus, ja jos tämä tapahtuu samassa ohjelmassa ei salauksella ole mitään vaikutusta, jos ulkopuolinen taho pääsisi ohjelmaan käsiksi. Vaihtoehtoista hälytyksen tapaa voisi tutkia lisää, sähköpostin sijaan sovelluksen teko ja käyttö hälytykseen voisi olla turvallisempi vaihtoehto.

Työn tietoturvaa saatiin parannettua ohjelmien ulkopuolella. Pelkästään vaihtamalla Raspberryn oletus-salasanan ja vaatimaan pääkäyttäjän komentojen suorittamiseen salasanan teemme Raspberryyn käsiksi pääsyn jo hieman vaikeammaksi. Tämä on tarpeen, jos Raspberry on jatkuvasti yhteydessä verkkoon, ja palomuurien käyttö on suositeltavaa.

Vaikka työssä käytettyjä ohjelmia voi kuvailla viattomiksi niiden toteutusta arvioidessa tämä ei tarkoita etteikö esineiden internetin tietoturvan tärkeys olisi tärkeää. Kuten aikaisemmin mainittiin, työn Raspberry oli jatkuvasti kiinni modeemissa. Olettaen että ulkopuolinen taho pääsisi saastuttamaan Raspberryn, ei olisi suuri harppaus saastuttaa itse modeemi ja kaikki siihen yhteyden ottavat laitteet, ainakaan teoriassa. Saastuneilla laitteilla voitaisiin esimerkkinä lähettää DDoS-palvelunestohyökkäyksiä joissa yksinkertaisimmillaan saastunut laite avaa useita yhteyksiä www-palveluun, ruuhkauttaen sen.

Loppuvuodesta 2017 98 maassa havaittiin palvelunestohyökkäyksiä, joista pisin kesti 215 tuntia. Parhaimmillaan hyökkäyksiä saattoi olla päivässä yli tuhat (Kaspersky Lab 2017). (liite 4)

Tietoturva on laaja aihealue ja tuskin mikään järjestelmä on immuuni ulkopuolisille hyökkäyksille. Raspberryn ja esineiden internetin tietoturvaan kannattaa mielestäni silti panostaa harrastelija- ja ammattilaistasolla. Toivoisin varsinkin korkean luokan yritysten panostamaan ennemmin tietoturvaan kuin ottamaan esineiden internetiä käyttöön ilman siihen tutustumista.

Olen lievästi pettynyt, ettei aikaa jäänyt haastavampien elektroniikan komponenttien liittämiseen työn osaksi. Alkuperäinen tavoite oli liittää työn kaksi osaa yhdeksi kokonaisuudeksi ja jopa laajentaa toimintaa erillisellä koodilla, liittäen uusia komponentteja, kuten sensoreita. Olen kuitenkin tyytyväinen työhön nykyisessä muodossaan, ja aikaa ylimääräiseen toteutukseen tuskin olisi jäänyt. Mielessäni on jatkoprojekteja, joissa voin käyttää vastaavia ohjelmia. Ensi sijalla on pyrkiä liittämään haastavampia komponentteja GPIO-liittimiin tai ottaa avuksi muita sulautettuja järjestelmiä.

Työtä tehdessä huomasin, että toteutustapoja Raspberryn etäkäyttöön on useita, tämä ei tullut varsinaisena yllätyksenä ottaen huomioon Raspberryn suosion harrastelijoiden keskuudessa. Eri toteutustapojen vertaaminen on näin myös itselleni yksi tulevaisuuden haasteista.

Kokonaisuudessaan työ eteni suunnitellussa aikataulussa pieniä viivästyksiä lukuun ottamatta ja lopputuloksena saatiin kaksi toimivaa esineiden internetin esimerkkiä, joilla on mahdollista ohjata komponenttien toimintaa verkosta käsin sekä siirtää ja vastaanottaa dataa eri laitteilla.

LÄHTEET

Adafruit 2017. Raspberry Pi Model B+ [viitattu 21.8.2017]. Saatavissa: https://www.adafruit.com/product/1914

Deveria A. 2017. Can I use Web Sockets? Compatibility table for support of Web Sockets in desktop and mobile browsers. [viitattu 18.9.2017]. Saatavissa: https://caniuse.com/websockets

Dataplicity 2017. [viitattu 8.9.2017]. Saatavissa: https://docs.dataplicity.com/v1.0/docs/

Google 2017. Third party apps. [viitattu 22.9.2017]. Saatavissa: https://support.google.com/plus/answer/2485911?hl=e

Kaspersky Lab 2017. DDoS attacks in Q3 2017 [viitattu 7.11.2017]. Saatavissa: https://securelist.com/ddos-attacks-in-q3-2017/83041/

Korpela J. 2009. Web-julkaisemisen opas. Saatavissa: http://jkorpela.fi/webjulk/

Michael Dory, Adam Parrish, Brendan Berg. 2012. Introduction to Tornado: Modern Web Applications with Python. Sebastopol: O'Reilly Saatavissa: http://www.youeer.com/wp-content/uploads/2012/04/OReilly.Introduction.to_.Tornado.Mar_.2012.pdf

Mouser Electronics 2017. 59045-010 Datasheet [viitattu 13.9.2017]. Saatavissa: http://www.mouser.com/ds/2/240/Littelfuse_Reed_Switches_59045_Datasheet.pdf-938246.pdf

Preeti Jain 2017. Reed Switch: Understanding Specifications. [viitattu 21.9.2017]. Saatavissa: https://www.engineersgarage.com/articles/reed-switch-specification

Python 2017. Documentation Contents. [viitattu 15.10.2017]. Saatavissa: https://docs.python.org/3.6/contents.html

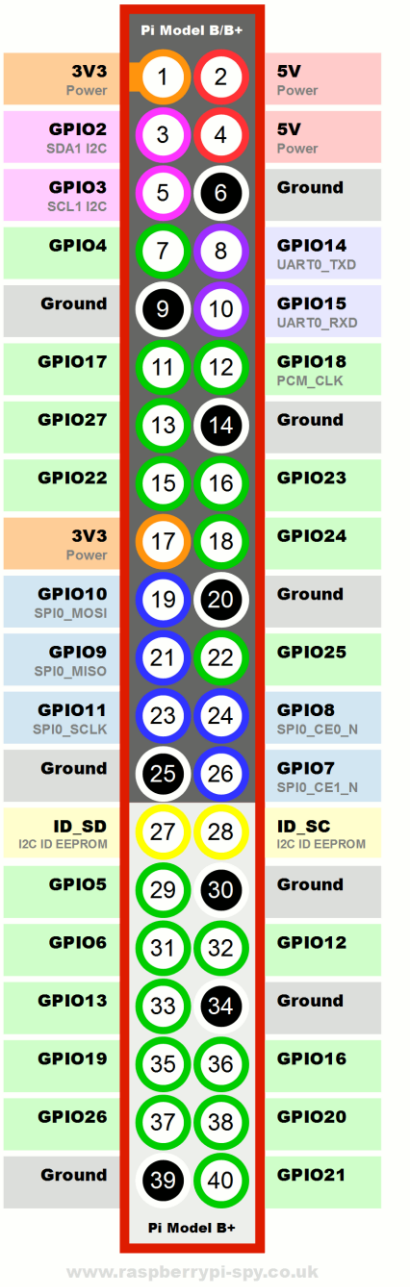
Raspberry Pi Foundation 2017 Documentation. [viitattu 6.9.2017]. Saatavissa: https://www.raspberrypi.org/documentation/

Raspberry Pi Spy 2012. Simple Guide to the RPi GPIO Header and Pins. Saatavissa: https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

Tornado 2017. Tornado 4.5.2 documentation. [viitattu 5.9.2017]. Saatavissa: http://www.tornadoweb.org/en/stable/

LIITTEET

LIITE 1. Raspberry Pi B+ GPIO-pinnit

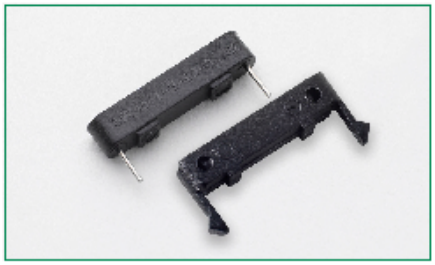


LIITE 2. Littlefuse 59045 Reed-kytkin

Littelfuse

Overmolded Reed Switches
Low Power > 59045

59045 Miniature PCB Mountable Reed Switch + 57045 Actuator



Description

The 59045 is a miniature PCB-mountable reed switch, 17.78mm x 4.32mm x 3.3mm (0.700" x 0.170" x 0.130") with a normally open contact. It has moulded stand-offs to allow board washing. It is capable of switching up to 200Vdc at 10W. It functions best with the matching actuator 57045-000.

Note: The 57045 Actuator is sold separately.

Features

- Two-part magnetically operated proximity switch
- Normally open contact configuration
- Moulded stand-offs to allow board washing
- Wave solder capable
- Certified for use in North American Hazardous Locations: Class I, Division 2 and Zone 2
- ATEX certified for use in European explosive atmospheres: II 3 G Ex nC IIC Gc

Agency Approvals

Agency	Agency File Number
	E61760 E471070
	DEMKO 14 ATEX 1393U II 3 G Ex nC IIC Gc

Note: Contact Littelfuse for specific agency approval ratings.

Benefits

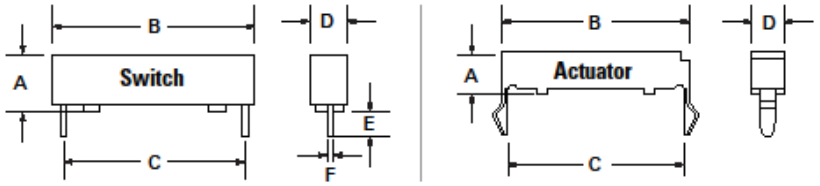
- Mounts directly into printed circuit board
- No standby power requirement
- Operates through non-ferrous materials such as wood, plastic or aluminium
- Hermetically sealed, magnetically operated contacts continue to operate long after optical and other technologies fail due to contamination

Applications

- Position and Limit Sensing
- Security System Switch
- Door Switch

Dimensions

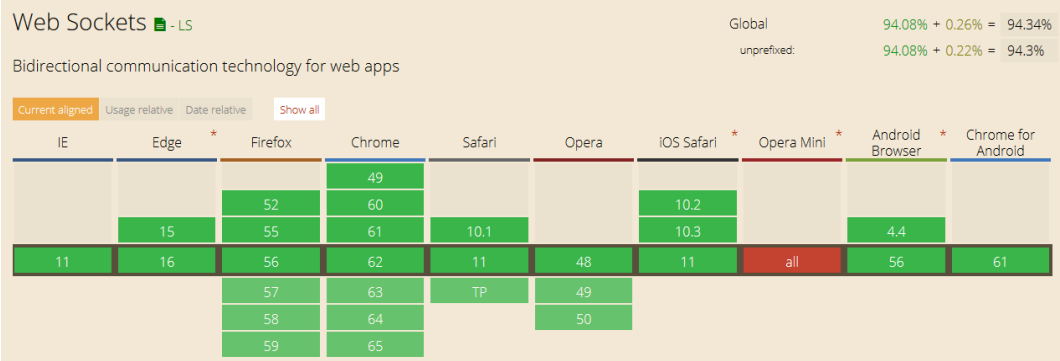
Dimensions in mm (inch)



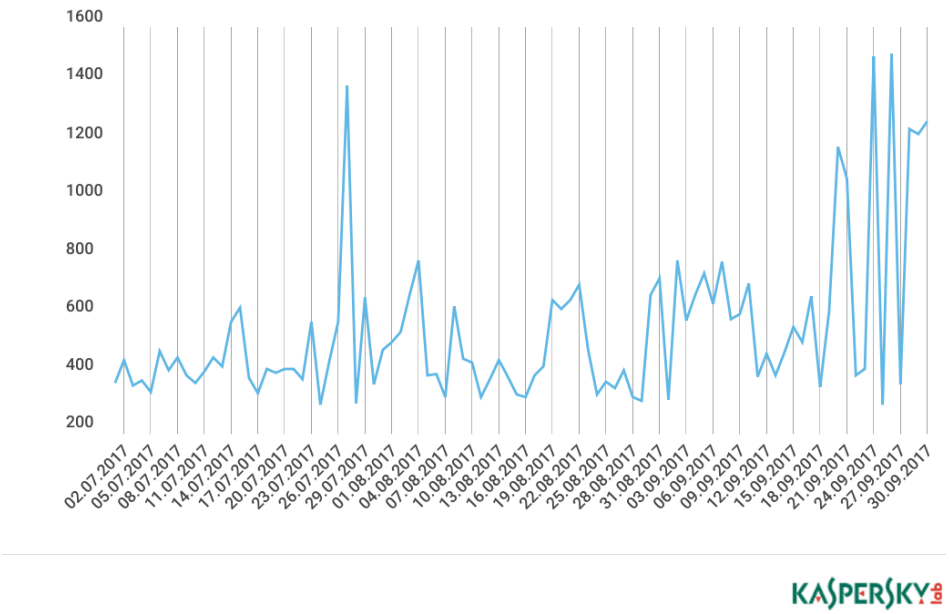
	A Max.	B Max.	C ±0.25 (.010)	D Max.	E ±0.38 (.015)	F Nom.
57045 Actuator	4.32 (.170)	17.78 (.700)	*	3.3 (.130)	–	–
59045 Switch	4.32 (.170)	17.78 (.700)	15.24 (.600)	3.3 (.130)	3.3 (.130)	0.4699 (.0185") Dia.

* Mounting: 1.57 (.062) thick board, 3.17 (.125) holes on 13.72mm (.540) centerline.

LIITE 3. WebSocketia tukevat selainversiot



LIITE 4. DDoS-hyökkäykset päivittäin Q3.2017



KASPERSKY lab